GESTIÓN

DE

CONFIGURACIÓN DEL SOFTWARE

CONCEPTOS BÁSICOS DE GESTIÓN DE CONFIGURACIÓN DEL SOFTWARE

El objetivo de esta unidad es introducir al alumno en la terminología y conceptos básicos que se manejan en la Gestión de Configuración del Software.

1.1. ¿Qué es la Gestión de Configuración del Software?

En términos muy generales, la Gestión de Configuración del Software (GCS) se puede definir como una disciplina cuya misión es controlar la evolución de un sistema software.

Según Babich, "El arte de coordinar el desarrollo de software para minimizar la confusión, se denomina Gestión de Configuración. La Gestión de Configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. El objetivo es maximizar la productividad minimizando los errores."

¿Qué papel juega la Gestión de Configuración dentro de un Proyecto de Desarrollo de Software? El éxito de un proyecto depende de la correcta ejecución de 4 tipos de funciones:

- Gestión del Proyecto, que incluye fundamentalmente la Estimación, Planificación y Seguimiento del proyecto, y la Organización, Dirección y Gestión de Recursos Humanos.

- Desarrollo Técnico: Actividades de Ingeniería del Software a lo largo de todo el ciclo de vida del producto (Análisis, Diseño, Codificación)

- Sistema de Calidad, que incluye las actividades de Validación (¿Estamos construyendo el producto correcto?), Verificación (¿Estamos construyendo el producto correctamente?) y Pruebas (¿Funciona el código?) y las actividades de Garantía de Calidad (Medidas encaminadas a asegurar que el producto se construye con unos determinados niveles de calidad).

- Sistema de Gestión de Configuración, aunque también se suele considerar a la Gestión de Configuración como una actividad de Garantía de Calidad.

constituyendo las dos últimas el Sistema de Control del Proyecto.

En resumen, la Gestión de Configuración es una disciplina de control, dentro del proyecto.

La Gestión de Configuración se debe realizar a lo largo de todo el ciclo de vida del producto, tanto en el desarrollo como en el mantenimiento, hasta que el producto se retira.

El objetivo de las actividades de Gestión de Configuración del Software es:

- establecer y mantener la integridad de los productos generados durante un proyecto de desarrollo de software y a lo largo de todo el ciclo de vida del producto.

- evaluar y controlar los cambios sobre ellos, es decir, controlar la evolución del sistema software.

- facilitar la visibilidad sobre el producto.

¿Qué significa la integridad de un producto software? Significa que el producto cumple las siguientes condiciones:

- Satisface las necesidades del usuario (cumple todos los requisitos del usuario, tanto los explícitos como los implícitos)

- Cumple los requisitos de rendimiento

- Se puede trazar su evolución desde que se concibió, y a través de todas las fases de su ciclo de vida.

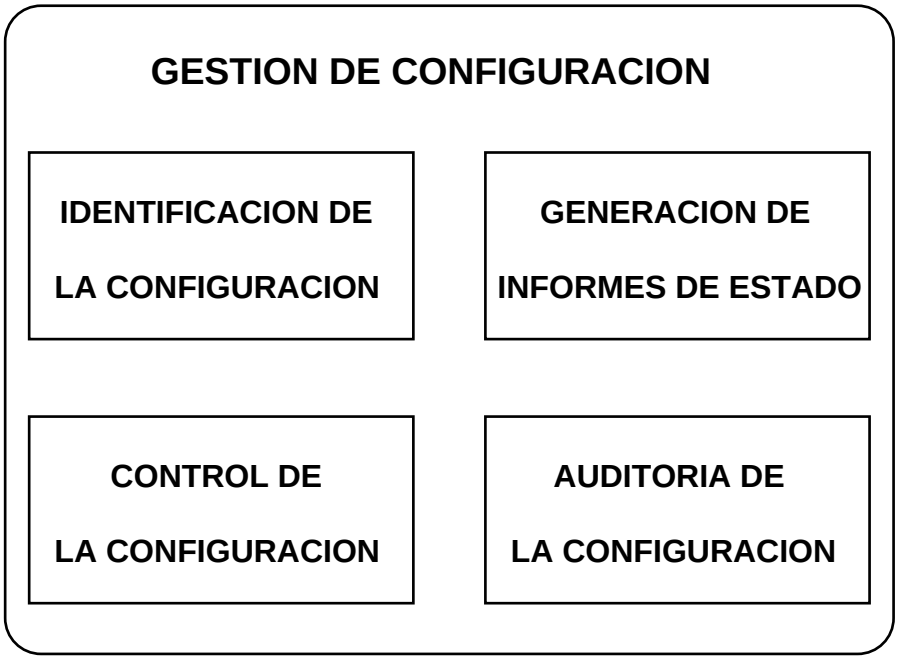
Para conseguir estos objetivos, la Gestión de Configuración plantea la realización de ciertas actividades. La definición estándar de Gestión de Configuración del Software, tal y como aparece en el estándar de IEEE, incluye las siguientes actividades:

- *Identificación de la Configuración*: Consiste en identificar la estructura del producto, sus componentes y el tipo de estos, y en hacerlos únicos y accesibles de alguna forma.

- *Control de Cambios en la Configuración*: Consiste en controlar las versiones y entregas de un producto y los cambios que se producen en él a lo largo del ciclo de vida.

- *Generación de Informes de Estado*: Consiste en informar acerca del estado de los componentes de un producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto.

- *Auditoría de la Configuración*: Consiste en validar la completitud de un producto y la consistencia entre sus componentes, asegurando que el producto es lo que el usuario quiere.



Sin embargo, los sistemas que automatizan la gestión de configuración suelen incluir algunas funciones adicionales, lo que nos lleva a ampliar la definición estándar con las siguientes actividades:

- Construcción: Consiste en gestionar la compilación y enlazado de los distintos componentes del producto software de una forma lo más eficiente posible.

- Control del Trabajo en Equipo: Consiste en controlar las interacciones que se producen entre los múltiples desarrolladores de un producto, sobre todo cuando deben compartir ciertos componentes del producto.

- Control de Versiones: Consiste en mantener un registro histórico de las diferentes versiones por las que pasan los componentes de un producto, que permita la recuperación de cualquiera de ellas.

- Gestión de Problemas: Consiste en realizar un seguimiento de la evolución de los problemas que afectan al producto.

Cada vez resulta más evidente que las necesidades de Gestión de Configuración en una organización grande, con aplicaciones software de larga vida, o que requieren del mantenimiento simultáneo de múltiples versiones, son muy grandes, y a veces pueden resultar muy complejas. Sin embargo, la mayor parte de las empresas de desarrollo de software, aún hoy en día, realizan una Gestión de Configuración “bajo mínimos”.

¿Por qué es compleja la Gestión de Configuración? Se pueden encontrar varias respuestas a esta pregunta:

- Número elevado de componentes a controlar: A medida que progresa el proceso de desarrollo de Software el número de componentes crece rápidamente.

- Pero el problema realmente surge porque en el proceso toma parte otra variable: el CAMBIO. Como dice la ley de la Ingeniería de Sistemas de Bersoff, “sin importar en qué momento del ciclo de vida del sistema nos encontremos, el

> sistema informático cambiará, y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida".

Hay que asumir el cambio, no evitarlo. Es importante tener en cuenta que la mayoría de los cambios están justificados, ya que a medida que pasa el tiempo se sabe más acerca del problema y de cómo resolverlo.

La complejidad de la Gestión de Configuración viene dada en gran medida por la falta de soluciones satisfactorias para otros problemas de la Ingeniería del Software. Así, por ejemplo, la Gestión de Configuración resultaría mucho más fácil si existiese alguna forma de describir la arquitectura de una aplicación que permitiese realizar cambios en el código y propagar sus efectos al resto del sistema de una manera sencilla.

La Gestión de Configuración está también fuertemente relacionada con el problema del mantenimiento del software. Sin una buena Gestión de Configuración, el mantenimiento de un producto puede ser una verdadera pesadilla.

Tiene también una gran influencia sobre otros aspectos del desarrollo de software, como son:

- Las metodologías: Las actividades de Gestión de Configuración deberán integrarse con la metodología de desarrollo que utilice la empresa. Su planificación dependerá de las fases que establezca la metodología, los productos que se generen, etc.

- El entorno de desarrollo de software: cómo de fuertemente integrada está la gestión de configuraciones con el resto de las capacidades del entorno de desarrollo, y qué tipo de funciones de Gestión de Configuración permite.

- El modelo de proceso software, puesto que afecta a la forma en que los desarrolladores hacen su trabajo, imponiendo políticas y procedimientos, y supervisando la forma en se realiza este trabajo. La Gestión de Configuración desempeña un papel importante a la hora de conseguir incrementar el nivel de madurez del proceso software de una organización. En concreto se trata de una de las áreas clave, en el modelo CMM (Capability Maturity Model), necesarias para que una organización de desarrollo de software alcance el nivel 2, de entre los 5 que plantea dicho modelo.

- La calidad del producto software, puesto que contribuye a mantener la integridad del producto.

- La organización, puesto que por lo general se suele implantar una solución global, que afecta a toda la organización, y requiere del establecimiento de nuevos roles y responsabilidades que deberán integrarse en la organización del proyecto.

1.2. ¿Qué es la Configuración del Software? ¿Y los Elementos de Configuración?

Al conjunto de toda la información y productos utilizados o producidos en un proyecto como resultado del proceso de Ingeniería de Software se le denomina configuración del software.

A cada uno de los componentes de la configuración del software se le llama elemento de configuración del software (ECS). El ECS es la unidad de trabajo para la GCS. El término configuración del software designa, por tanto, el conjunto de todos los elementos de configuración del software de un proyecto.

Se pueden considerar como Elementos de Configuración del Software los siguientes componentes, entre otros:

1. La especificación del sistema.

2. El plan del proyecto software.

3. La especificación de requisitos software.

4. Un prototipo, ejecutable o en papel.

5. El diseño preliminar.

6. El diseño detallado.

7. El código fuente.

8. Programas ejecutables.

9. El manual de usuario.

10. El manual de operación e instalación.

11. El plan de pruebas.

12. Los casos de prueba ejecutados y los resultados registrados.

13. Los estándares y procedimientos de ingeniería de software utilizados.

14, Los informes de problemas.

15. Las peticiones de mantenimiento.

16. Los productos hardware y software utilizados durante el desarrollo.

17. La documentación y manuales de los productos hardware y software utilizados durante el desarrollo.

18. Diseños de bases de datos.

19. Contenidos de bases de datos.

En cualquier caso, para cada proyecto concreto se debe determinar qué se va a considerar como ECS.

Muchas organizaciones también ponen los productos hardware y software utilizados durante el desarrollo bajo control de configuración. Como estos productos son necesarios para desarrollar el software, deberán estar disponibles cuando se realicen cambios sobre su configuración. Aunque no es normal que haya problemas, es posible que una nueva versión de una herramienta produzca resultados diferentes que la versión usada (un compilador, por ejemplo).

Un ECS debe ser un elemento que se pueda definir y controlar de forma separada. Es decir, debe ser una unidad en sí mismo.

En cuanto al software propiamente dicho, dependiendo de su tamaño, complejidad y necesidad de control y visibilidad sobre el mismo, puede requerir de su descomposición en varios ECS, aunque el sistema en su conjunto será a su vez un ECS.

A medida que progresa el proceso de desarrollo, el número de ECS crece rápidamente. La Especificación del Sistema da lugar al Plan del Proyecto y a la Especificación de Requisitos Software. A su vez estos dan lugar a otros documentos, etc. Si simplemente cada ECS produjera otros ECS, no habría prácticamente confusión. El problema aparece cuando entra en juego la variable CAMBIO.

1.3. Líneas base

Como ya hemos visto, uno de los objetivos principales de la Gestión de Configuración va a ser el de gestionar los cambios que se producen en el sistema a lo largo de su ciclo de vida.

Para controlar los cambios sin impedir los cambios justificados se utiliza el concepto de LÍNEA BASE.

Se puede definir una línea base como un punto de referencia en el proceso de desarrollo del software que queda marcado por la aprobación de uno o varios Elementos de Configuración del Software, mediante una revisión técnica formal.

También se puede definir como un elemento que ha sido revisado y aceptado, que va a servir como base para otros desarrollos posteriores y que solamente puede cambiarse a través de un proceso formal de control de cambios.

La idea consiste en permitir cambios rápidos e informales sobre un Elemento de Configuración del Software antes de que se pase a formar parte de una línea base, pero en el momento en que se establece una línea base se debe aplicar un procedimiento formal para evaluar y verificar cada cambio.

1.4. Versiones, Revisiones, Variantes, Configuraciones Alternativas y Releases

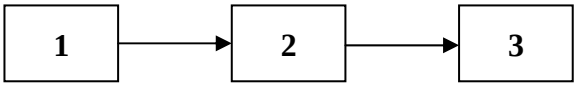
Se puede definir una VERSIÓN como una instancia de un elemento de configuración, en un momento dado del proceso de desarrollo, que es almacenada en un repositorio, y que puede ser recuperada en cualquier momento para su uso o modificación.

A las distintas versiones que aparecen en el tiempo, según se va avanzando en el desarrollo de un elemento, se les suele llamar también REVISIONES.

Entre las distintas revisiones de un elemento de configuración se pueden establecer relaciones de sucesión temporal. Una representación posible para las diferentes revisiones de un elemento y sus relaciones de sucesión es mediante un GRAFO DE EVOLUCIÓN o grafo de revisión.

La manera más fácil de crear una nueva revisión de un ECS es realizar una modificación sobre la revisión más reciente. De esta manera las revisiones van formando una cadena, a la que se suele llamar CADENA DE REVISIÓN. Cada revisión en la cadena es una actualización de, y viene a sustituir a la revisión anterior. Normalmente se trabajará sobre la última revisión de la cadena, que es la más actual, aunque las anteriores también deben ser accesibles.

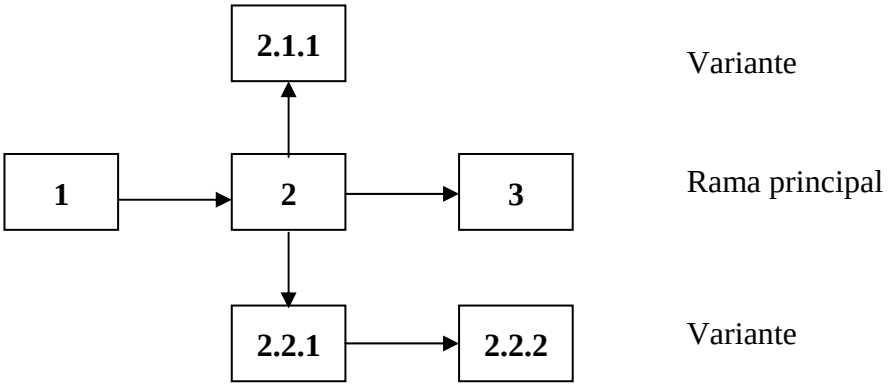
Cada una de las revisiones de un ECS se debe poder identificar de manera única, siendo común utilizar un esquema numérico, donde cada nueva versión recibe un número sucesivo.



Existen herramientas de Control de Versiones que se encargan de la identificación de las distintas versiones de cada elemento, de mantener información acerca de las relaciones que hay entre ellas y de su almacenamiento en un repositorio.

La Gestión de Configuración debe permitir también especificar y gestionar distintas VARIANTES de los elementos de configuración. Las variantes son versiones de un elemento de configuración que coexisten en un determinado momento y que se diferencian entre sí en ciertas características. Las variantes representan la necesidad de que un objeto satisfaga distintos requisitos al mismo tiempo. A diferencia con las revisiones, que son estrictamente secuenciales, y sólo existe una como revisión actual, las variantes se desarrollan en paralelo, y puede haber varias sobre las que se esté trabajando simultáneamente.

Una variante no reemplaza a otra, como ocurre con las revisiones, sino que abre un nuevo camino de desarrollo. Las variantes se reconocen fácilmente en el grafo de evolución, puesto que aparecen como una ramificación de éste.



A su vez, una variante puede pasar por una sucesión de revisiones, que deberán ser identificadas convenientemente.

Las variantes pueden ser temporales o permanentes. Variante temporal es toda aquella que se acabará mezclando con otra variante en algún momento del desarrollo. En ciertas ocasiones es necesario que varias personas trabajen simultáneamente sobre la misma versión de un objeto, y para que no ocurran conflictos entre ellas, se crea una variante distinta para cada persona, para que puedan trabajar en paralelo. Una vez que todas ellas han acabado las modificaciones, es necesario mezclar todas las variantes para que la versión resultante contenga todos los cambios realizados. Las variantes temporales deben evitarse, y en todo caso mezclarse lo antes posible, ya que a medida que pasa el tiempo divergerán más de la versión original y entre sí, y resultará más difícil su fusión.

Otra posibilidad que nos puede llevar a la creación de variantes es cuando no somos capaces de decidir qué alternativa de las que se nos presentan es la más adecuada. Una posible solución es desarrollar en paralelo distintas variantes para explorar las diferentes soluciones alternativas, y luego quedarnos con la más conveniente. En este caso no es necesario fundir variantes, sino que las variantes descartadas quedan abandonadas. Son variantes experimentales.

Otras variantes “de usar y tirar” son las Variantes de Pruebas: Son variantes sobre las que se introducen elementos especiales para facilitarnos la realización de pruebas, por ejemplo para permitirnos trazar el flujo de control de un proceso o para comprobar el contenido y evolución de las estructuras de datos internas.

Por otro lado, las variantes permanentes aparecen para dar respuesta a conjuntos distintos de requisitos, y tampoco se llegarán a mezclar entre sí. Podemos dividir este tipo de variantes en dos categorías:

- Variantes de requisitos de usuario: Tenemos distintos usuarios con distintos requisitos. El caso más típico es el idioma usado por estos.

- Variantes de plataforma: Debemos realizar una variante por cada sistema operativo o plataforma hardware sobre la que deseemos que funcione nuestra aplicación.

El hecho de que se abran ramificaciones en el grafo de evolución hace que ya no se pueda hablar sencillamente de “la configuración” de un sistema. En vez de una única configuración vamos a tener un conjunto de CONFIGURACIONES ALTERNATIVAS. Cada configuración alternativa va a satisfacer las necesidades de un entorno particular o

usuario, y se debe especificar mediante la selección de los ECS que la componen y de la versión adecuada de cada uno de ellos. Esto se puede conseguir de la siguiente forma:

- En primer lugar se asocian atributos a cada versión de un ECS. Estos atributos pueden ser simplemente el número de versión asociado a cada objeto, o pueden ser tan complejos como una cadena de pares (atributo,valor) unidos con operadores booleanos. A estas expresiones lógicas que representan conocimiento acerca de versiones y componentes se les llama CARACTERÍSTICAS LÓGICAS. Entre las herramientas que utilizan sólo atributos numéricos está SCCS, y entre las que usan atributos simbólicos está RCS, que es una evolución de SCCS. Ejemplos de herramientas que permiten usar características lógicas son NSE o DSEE.

- En segundo lugar, se crea una especificación de configuración que describa el conjunto de atributos deseado, y se recuperan los ECS adecuados para construir la configuración.

Un ejemplo puede ser un programa que tenga una configuración para monitores en color, que incluya los módulos 1,2,3 y 4, y otra configuración para monitores monocromos que incluya los módulos 1,2,3 y 5. Se podría entonces asociar el atributo "color" a aquellos componentes que se deben incluir en la configuración para color y el atributo "monocromo" a los que se deben incluir en la configuración para monocromo.

Una vez que se han especificado las diferentes configuraciones del producto, existen herramientas que facilitan la construcción automática de una configuración concreta, es decir, la recuperación de los ECS necesarios en la versión adecuada, su compilación y montaje. Un ejemplo de este tipo de herramientas es MAKE.

MAKE usa un fichero llamado Makefile para definir la configuración. El usuario define en este fichero la especificación de cómo los componentes de un sistema están relacionados entre sí y cómo procesarlos para crear una configuración actualizada. A la hora de construir la configuración MAKE revisa las fechas en las que los componentes fueron modificados por última vez, buscando así la cantidad mínima de componentes a compilar para generar la nueva configuración. Su principal problema es que no hay manera de decirle a MAKE que utilice una u otra versión. Por otro lado, MAKE no es una herramienta de control de versiones, por lo que debe usarse en combinación con una de ellas.

Se suele llamar RELEASE a una configuración del sistema que se va a comercializar o entregar al cliente.

1.5. Almacenamiento de Versiones

Una posibilidad es almacenar únicamente la última versión de cada uno de los ECS del sistema, pero esto puede plantearnos numerosos problemas.

A veces se tiene que versiones anteriores siguen formando parte de sistemas que están todavía en uso, por lo que hay que seguir manteniéndolas, a pesar de tener versiones nuevas más actualizadas.

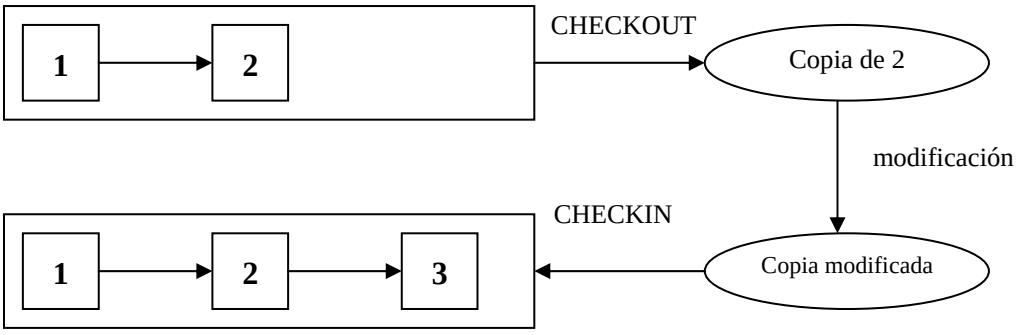
Otras veces puede ser conveniente guardar versiones anteriores como punto de seguridad, al cual poder volver en caso de que un cambio efectuado no nos lleve al resultado deseado.

También puede ser conveniente guardar versiones antiguas para poder reutilizar elementos que aparecían en ellas pero que fueron desechados en un momento dado.

Las herramientas de control de versiones o gestión de revisiones nos ayudan a crear, identificar y almacenar nuevas versiones, al mismo tiempo que se mantienen las versiones anteriores.

En cuanto a la creación de nuevas versiones, un extremo consiste en que toda modificación, por pequeña que sea, conduzca a la creación de una nueva versión. El problema que tiene esta opción es que el número de revisiones crece desmesuradamente, y puede llegar a ser imposible mantenerlas todas. Otra opción es dejar que sea el usuario el que decida cuándo se debe crear una nueva versión. El desarrollador puede modificar un objeto tantas veces como desee, manteniéndose el mismo identificador de versión, hasta que decida que dicha versión debe ser CONGELADA. En este momento la versión se almacena y ya no puede volver a ser modificada. Si se necesita realizar nuevas modificaciones se debe crear una nueva versión, sobre la que se trabajará hasta que sea congelada.

El modelo de trabajo sobre el que se basan la mayoría de las herramientas de gestión de revisiones es el de la figura:



Todas las revisiones se encuentran almacenadas en un repositorio gestionado por la herramienta. Si alguien necesita trabajar en uno de los objetos que se encuentran en el repositorio, debe realizar un CHECKOUT. Con esto está solicitando una copia de la última versión del objeto que se encuentre en el repositorio, y esta copia se almacenará en un directorio de trabajo local. A partir de este momento la copia queda fuera del alcance de la herramienta y puede ser modificada por el usuario. Una vez acabado el cambio, la copia es almacenada de nuevo en el repositorio mediante una operación llamada CHECKIN, que insertará la copia en el repositorio como una nueva versión sucesora de la que se extrajo de él.

Además, las herramientas de control de versiones utilizan maneras más eficientes para almacenar las versiones, para disminuir la cantidad de espacio requerido. Normalmente estas herramientas no almacenan físicamente todas las versiones, sino solamente una de ellas, que puede ser la primera o la última. Sin embargo, nos permiten recuperar cualquier otra versión. Para ello guardan también toda la historia de cambios que han ocurrido sobre el elemento y que lo han hecho pasar de una versión a otra.

Siendo r1 y r2 dos revisiones consecutivas en el grafo de evolución, llamamos DELTA a una secuencia de operaciones que aplicadas sobre la revisión r1 dan como resultado la revisión r2:

R2 = aplicar (R1, DELTA)

Si lo que se almacena en la herramienta es la primera versión, lo que se guardan son los llamados DELTAS DIRECTOS, que son los cambios que hacen pasar de una versión a la siguiente. Si lo que se almacena es la última versión, lo que se guardan son los llamados DELTAS INVERSOS, que son los cambios que hacen pasar de una versión a la anterior. La ventaja de guardar la última versión es que normalmente es la versión sobre la que se está trabajando, y su recuperación es más rápida. Para recuperar cualquier otra versión será necesario deshacer los cambios, aplicando los deltas inversos necesarios, y esto lleva un tiempo.

Hay herramientas que almacenan los deltas en ficheros separados del documento original, mientras que hay otras que los almacenan dentro del mismo documento. Son los llamados DELTAS MEZCLADOS.

Vamos a ver un ejemplo. Suponiendo que tenemos la revisión 1.1 con el siguiente contenido:

```
PROCEDURE Fac (Input: CARDINAL; VAR Result: CARDINAL);
BEGIN
IF Input = 0 THEN
   Result := 1;
ELSE Fac (Input-1), Result);
     Result := Input * Result;
END
END Fac;
```

Tras una serie de modificaciones el resultado que obtenemos es el siguiente, que corresponderá a la revisión 1.2:

```
PROCEDURE Fac (Input: CARDINAL): CARDINAL);
BEGIN
IF Input = 0 THEN
   RETURN 1;
ELSE RETURN Fac (Input-1) * Input;
END
END Fac;
```

Veamos cómo sería el delta mezclado, utilizando la siguiente notación:

I <n>	Incluir lo siguiente en la revisión n
D <n>	Borrar lo siguiente en la revisión n
E <n>	Final de un bloque de tipo I o D para la revisión n

```
I1.1
D1.2
   PROCEDURE Fac (Input: CARDINAL; VAR Result: CARDINAL);
E1.2
I1.2
   PROCEDURE Fac (Input: CARDINAL): CARDINAL);
E1.2
   BEGIN
   IF Input = 0 THEN
D1.2
   Result := 1;
E1.2
I1.2
   RETURN 1;
E1.2
D1.2
   ELSE Fac (Input-1), Result);
        Result := Input * Result;
E1.2
I1.2
   ELSE RETURN Fac (Input-1) * Input;
E1.2
   END
   END Fac;
E1.1
```

Y ahora veamos cómo sería el delta inverso separado, utilizando la siguiente notación:

Del <n>	Borrar la línea n
Add <n1> <n2>	Añadir en la línea n1 un total de n2 filas

Que nos permitiría reconstruir la versión 1.1. a partir de la 1.2. Lo primero que hay que hacer es numerar las líneas del fichero donde se almacena la versión 1.2.

```
1 PROCEDURE Fac (Input: CARDINAL): CARDINAL);
2 BEGIN
3 IF Input = 0 THEN
4    RETURN 1;
5 ELSE RETURN Fac (Input-1) * Input;
6 END
7 END Fac;
```

Y el delta inverso de 1.2 a 1.1 quedará:

```
Del 1
Add 1,1
   PROCEDURE Fac (Input: CARDINAL; VAR Result: CARDINAL);
Del 4
Add 4,1
   Result := 1;
Del 5
Add 5,2
   ELSE Fac (Input-1), Result);
        Result := Input * Result;
```

Algunas de las herramientas de control de versiones más conocidas son SCCS, RCS, CVS, RCE, PRCS, DSEE, PVCS, ClearCase, SourceSafe, etc.

2. IDENTIFICACIÓN DE LA CONFIGURACIÓN

En esta unidad se analizará con más detalle la primer actividad de las cuatro tareas básicas de gestión de configuración, la Identificación de la Configuración.

Esta tarea consiste en identificar y asignar nombres significativos y consistentes a todos y cada uno de los elementos que forman parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software.

La tarea de Identificación engloba o puede englobar varias actividades:

- Establecimiento de una jerarquía preliminar del producto software
- Selección de los Elementos de Configuración
- Definición de las relaciones en la configuración
- Definición de un Esquema de Identificación
- Definición y Establecimiento de líneas base.
- Definición y Establecimiento de bibliotecas de software.

A continuación veremos en más detalle cada una de estas actividades.

2.1. Establecimiento de una jerarquía preliminar del producto software

Se trata de obtener una primera visión de la estructura y los elementos que tendrá el sistema software. Esta jerarquía facilitará la ejecución de otras actividades posteriores de GC, como la selección de los elementos de configuración o la asignación de números de identificación a los documentos. Se realizará, opcionalmente, en los primeros compases del proyecto.

2.2. Selección de los Elementos de Configuración

La selección de un número adecuado de ECS es muy importante. El tener demasiados puede provocar un número excesivamente elevado de especificaciones y documentos que al final resulta inmanejable. Otro problema es que mantener un EC es costoso, ya que puede requerir:

- Especificación independiente
- Plan de pruebas
- Manuales independientes
- Revisión por parte del usuario
- Necesidad de aprobación de los cambios importantes por parte del usuario.
- Auditorías funcionales y físicas independientes.
- Número de identificación separado.
- Contabilidad de estado separada.
- Trazabilidad de su evolución.

Sin embargo, el tener pocos ECS puede hacer que no se tenga la suficiente visibilidad sobre el producto.

Al seleccionar los ECS hay que tratar de separar en ECS distintos las funcionalidades distintas, para tratar de minimizar el impacto de los cambios. Si un ECS incluye funciones muy diversas es probable que aumente el número de veces que será necesario repetir el proceso de reconstrucción del elemento y liberación de una nueva versión.

Algunos criterios adicionales que se pueden utilizar para la selección de ECS son los siguientes:

- Utilización múltiple: Número de elementos de su mismo nivel o niveles superiores que lo utilizan.
- Criticidad: Gravedad del impacto de un fallo en dicho componente.
- Número de personas implicadas en su mantenimiento.
- Complejidad de su interfaz: Las interfaces de un EC con otros deberían ser simples. Hay que minimizar el acoplamiento entre ECs.
- Singularidad del componente con respecto al resto.
- Reutilización: Si el componente se va a diseñar especialmente para ser reutilizado.
- Si se trata de elementos reutilizados.

- Tipo de tecnología: Si el componente incorpora nuevas tecnologías no utilizadas en otros componentes.

En cualquiera de estos casos, el componente que se analiza tendrá más probabilidades de convertirse en un ECS, puesto que requerirá de una mayor atención.

2.3. Definición de relaciones en la configuración

Se puede considerar que los ECS son objetos, y están conectados con otros ECS mediante relaciones.

Esta información ayudará al personal de GCS a comprender dónde se situa un elemento con respecto al resto. Hay que tener en cuenta que el personal de GCS suele ser externo al equipo de desarrollo y necesita este tipo de ayudas para poder asomarse a los productos y el proceso de desarrollo.

Algunas de las relaciones que puede ser interesante mantener son las siguientes:

- Equivalencia: Por ejemplo, cuando un determinado ECS que es un programa está almacenado en tres lugares diferentes (un disco maestro, una copia de seguridad en cinta y el diskette del programador), pero todas las copias corresponden al mismo programa.

- Composición: Por ejemplo, el ECS “especificación de diseño” estará compuesto de otros ECS, como el “modelo de datos” o el “diseño del módulo N”, para cada uno de los módulos que componen el producto software.

- Dependencia: Cualquier otro tipo de relaciones entre ECS, fundamentalmente en la documentación, y sobre todo para facilitar la trazabilidad de los requisitos. Así, por ejemplo, los objetos tienen dependencia del modelo de clases.

- Derivación: A partir de qué se ha originado algo. Por ejemplo, el código objeto del código fuente, o una determinada traza de ejecución de un determinado caso de prueba con un determinado programa ejecutable.

- Sucesión: En la historia de cambios sobre un elemento desde una revisión a otra. Puede ser muy útil definir un Grafo de Evolución para cada ECS. Este grafo describe la historia de cambios de un objeto y su transición de unas versiones a otras.

- Variante: Variación sobre un determinado elemento, con la misma funcionalidad, pero que, por ejemplo, funciona más rápido.

Gracias a estas relaciones, si se lleva a cabo un cambio sobre un ECS, se podrá determinar fácilmente qué otros ECS pueden verse afectados.

Todas estas interrelaciones se pueden representar con un lenguaje de interconexión de módulos (LIM). Una de sus utilidades es que nos permitirá construir automáticamente cualquier versión de un sistema.

2.4. Definición de un Esquema de Identificación

Es necesario decidir también cuál es el método que se va a utilizar para identificar de forma unívoca cada Elemento de Configuración del Software. Dicho en otras palabras, es necesario establecer un Esquema de Identificación que permita etiquetar cada uno de los Elementos de Configuración del Software. Sea cual sea, el esquema de identificación utilizado debe proporcionar al menos la siguiente información:

1. Número o código del Elemento de Configuración del Software.
2. Nombre del ECS
3. Descripción del ECS
4. Autor/es del ECS
5. Fecha de creación
6. Identificación del proyecto al que pertenece el Elemento de Configuración del Software.
7. Identificación de la línea base a la que pertenece.
8. Identificación de la fase y subfase en la que se creó.
9. Tipo de Elemento de Configuración (documento, programa, elemento físico (cintas, discos, etc.),...).
10. Localización

Por otro lado, el esquema de identificación debe permitir diferenciar entre las diferentes versiones de un mismo Elemento de Configuración del Software, puesto que los Elementos de Configuración del Software van a evolucionar a lo largo del ciclo de vida.

Para ello, se suelen utilizar los siguientes campos:

11. Número de versión
12. Fecha de la versión

Toda esta información puede ir codificada y agrupada en un único código de identificación o puede ser referenciada como partes separadas. A la hora de definir un código, se puede elegir entre dos tipos de sistemas de codificación:

- significativos
- no significativos (por ejemplo, asignar números consecutivos).

Una codificación no significativa requiere del establecimiento y mantenimiento de un registro de asignación de códigos. Aunque es más sencilla la selección del código, tiene la desventaja de que el código no da ninguna información acerca del tipo de ECS de que

se trata, o cualquier otra. Una codificación significativa puede ser muy simple o muy compleja, dependiendo del tipo y cantidad de información que codifica acerca del ECS. La ventaja es que no es necesario mirar en el registro de asignación de códigos para saber de qué ECS se trata. En cualquier caso, si la codificación es tan compleja que los que la usen no van a ser capaces de recordar el significado de los códigos, no tiene sentido. Acabarán consultando el registro de asignación de códigos. Hay que elegir una numeración significativa sólo si ello va a ser útil.

Los datos de identificación se pueden mantener en una base de datos automatizada, lo que permitirá, por ejemplo, referenciar fácilmente todos los Elementos de Configuración del Software de una determinada versión. Las herramientas para gestión automática de la configuración suelen ofrecer facilidades de identificación para los diferentes componentes software del producto (librerías de programas), aunque muchas de ellas no soportan la identificación de otros tipos de Elementos de Configuración, como los documentos o dispositivos físicos.

2.5. Definición y Establecimiento de líneas base

El concepto clave para realizar esta actividad es el de Línea Base. Las líneas base se establecen en hitos previamente especificados a lo largo del proceso de desarrollo (“milestones”).

Uno de los primeros pasos para poder efectuar la actividad de Identificación de la Configuración, por lo tanto, consiste en definir cuáles van a ser los hitos, dentro del proceso de desarrollo, en los que se va a establecer una línea base.

Lo más corriente es establecer líneas base al finalizar determinadas fases del ciclo de vida de desarrollo, con dos objetivos:

- Identificar los resultados de las tareas realizadas durante la fase.

- Asegurar que se ha completado la fase.

Aunque se pueden definir las líneas base con cualquier nivel de detalle, las líneas base más comunes son:

- Línea Base Funcional, que se establece al finalizar la fase de análisis y especificación de los requisitos del sistema, y comprende todos aquellos documentos en los que se define el problema a resolver, los costes del proyecto, el plan de tiempos, y los diferentes requisitos funcionales, de interoperatividad y de interfaz del sistema.

- Línea Base de Distribución o Asignación de funciones, que se establece al finalizar la fase de análisis y especificación de requisitos software, y comprende toda la documentación que gobernará el desarrollo de cada uno de los componentes software que se han identificado en la especificación del sistema, y la asignación o reparto de las diferentes funciones entre los distintos componentes del sistema

- Línea Base de Diseño Preliminar, que se establece al finalizar la fase de diseño preliminar. Comprende todos aquellos documentos en los que se define la arquitectura del producto software, así como el Plan de Pruebas.

- Línea Base de Diseño, que se establece al finalizar la fase de diseño detallado. Comprende todos aquellos documentos que contienen el diseño detallado del software y el plan de implementación, y también la descripción de los casos de prueba.

- Línea Base de Producto, que se establece al finalizar la fase de pruebas. Comprende los programas creados y todos aquellos documentos que contienen la información relativa a las pruebas realizadas.

- Línea Base de Operación, que se establece al finalizar la fase de implantación. Comprende los manuales de usuario, guías de operación y mantenimiento, manuales de formación, etc.

Hay autores que sólo consideran las líneas base Funcional, de Asignación, y de Producto, más una Línea Base de Desarrollo, que va evolucionando a lo largo del desarrollo, y que incluiría la Línea Base de Diseño Preliminar y la Línea Base de Diseño.

Una línea base se puede establecer de dos formas:

-FÍSICAMENTE: Etiquetando cada Elemento de Configuración del Software y almacenándolos en un Archivo o Biblioteca de Proyecto.

-LÓGICAMENTE: Publicando un documento de identificación de la configuración, que identifica el estado actual del producto en dicho punto del proceso de desarrollo.

En cualquier caso, al comienzo de cada nuevo proyecto se debería determinar de forma precisa:

- cuál es el ciclo de vida a utilizar,

- cuáles son sus fases,

- cuáles son los productos de cada fase,

- en qué puntos se van a establecer líneas base y

- cuáles van a ser los Elementos de Configuración del Software que contenga cada línea base.

2.6. Definición y Establecimiento de bibliotecas de software

Una biblioteca de software es una colección controlada de software y/o documentación relacionada cuyo objetivo es ayudar en el desarrollo, uso o mantenimiento del sofware.

Facilitan la tarea de GCS, especialmente en cuanto al Control de Cambios y la Contabilidad de Estado.

Se suelen establecer los siguientes tipos de bibliotecas de software o áreas:

- Biblioteca de trabajo: Se establece al inicio del proyecto, y comprende el área de trabajo donde los analistas y diseñadores elaboran los documentos del proyecto y donde los programadores desarrollan el software, es decir, donde se realiza la codificación y pruebas unitarias. Una vez se han completado las revisiones o pruebas y el elemento de configuración en cuestión ha sido revisado y aprobado, se inicia la transferencia del ECS a la Biblioteca de Soporte al Proyecto. En esta biblioteca el control de cambios es informal.

- Biblioteca de Integración. Es de esta biblioteca de donde se toman los ECS para su integración en ECS de nivel superior del sistema.

- Biblioteca de Soporte al Proyecto: En esta biblioteca se almacenan los ECS aprobados y transferidos desde la Biblioteca de Trabajo o desde la Biblioteca de Integración. Cuando un elemento pasa a esta biblioteca, se encuentra sujeto a un control de cambios interno y semiformal.

- Biblioteca de Producción: Está compuesta por la Biblioteca de trabajo, la de integración y la Biblioteca de Soporte al Proyecto

- Biblioteca maestra: Se usa para almacenar ECS liberados para su entrega al cliente o distribución en el mercado. Los elementos en la biblioteca maestra se encuentran sujetos a un control de cambios formal y estricto. Y normalmente esta biblioteca tiene fuertes restricciones de acceso para escritura, aunque normalmente no los tiene para lectura. En esta biblioteca se almacenan las Releases del sistema.

- Repositorio de Software: Es la entidad en la que se archivan los ECS de un proyecto tras su cierre. Se transfieren a él desde la Biblioteca Maestra. Opcionalmente se puede identificar un segmento especial en el que se almacenarán los elementos reutilizables. Todo lo que se almacena en el repositorio debe estar adecuadamente identificado y catalogado, para facilitar su recuperación en caso de necesidad. Se supone que es un almacenamiento a largo plazo, por lo que puede ser de recuperación lenta (cinta). Es central y común a todos los proyectos, mientras que la biblioteca de Producción y la Maestra son individuales para cada proyecto.

- Biblioteca de Backup: También debe estar adecuadamente identificada, aunque su contenido no está sujeto a Gestión de Configuración (las copias contenidas en ella no están catalogadas en los registros de Gestión de Configuración).

Un ECS defectuoso podría volver en cualquier momento a la Biblioteca de Trabajo para ser modificado y generar una nueva versión.

La tarea de bibliotecario suele estar encomendada a uno de los miembros del proyecto.

Además de elegir el tipo de Bibliotecas de Software que se van a mantener, se deben definir los procedimientos para:

- El establecimiento de una Biblioteca.

- La introducción de elementos en la biblioteca.

- El acceso a la biblioteca

3. CONTROL DE CAMBIOS EN LA CONFIGURACIÓN

Es la actividad de Gestión de Configuración más importante y su objetivo es proporcionar un mecanismo riguroso para controlar los cambios, partiendo de la base de que los cambios se van a producir. Normalmente combina procedimientos humanos y el uso de herramientas automáticas.

Se pueden considerar fundamentalmente dos tipos de cambios:

- Corrección de un defecto: Los clientes tienden a clasificar todos los cambios en esta categoría.

- Mejora del sistema: Los programadores, sin embargo, los suelen clasificar aqui.

Para solucionar el problema de determinar realmente de qué tipo es un cambio es importante la trazabilidad de los requisitos.

Por lo general se establecen varios niveles de control de cambios:

- Control de cambios informal: Antes de que el Elemento de Configuración del Software pase a formar parte de una línea base, aquel que haya desarrollado el Elemento de Configuración del Software podrá realizar cualquier cambio justificado sobre él.

- Control de cambios al nivel del proyecto o semi-formal: Una vez que el Elemento de Configuración del Software pasa la revisión técnica formal y se convierte en una línea base, para que el encargado del desarrollo pueda realizar un cambio debe recibir la aprobación de:

  - El director del proyecto, si es un cambio local

  - El Comité de Control de Cambios, si el cambio tiene algún impacto sobre otros Elementos de Configuración del Software

- Control de cambios formal: Se suele adoptar una vez que se empieza a comercializar el producto, cuando se transfieren los ECS a la Biblioteca

Maestra. Todo cambio deberá ser aprobado por el Comité de Control de Cambios.

En un proceso formal o semi-formal, aparece una nueva figura en la Organización, el Comité de Control de Cambios. El Comité de Control de Cambios es una persona o grupo encargado de tomar las decisiones finales acerca del estado y la prioridad de las peticiones de cambio.

Su obligación es tener una visión general del producto para poder evaluar el impacto de cada cambio en un determinado Elemento de Configuración sobre otros Elementos de Configuración, así como el impacto sobre la calidad del producto, su rendimiento, su fiabilidad, la visión que el cliente tiene del producto, etc. Esta labor se suele delegar en el Comité de Revisión del Diseño.

Dependiendo del tamaño del proyecto y de la empresa estará integrado por una o varias personas.

De todas formas, no sólo el Comité de Control de Cambios es responsable del Control de Cambios, también van a tener alguna responsabilidad:

- Todos los miembros de un proyecto:
  - pueden solicitar cambios
  - serán los encargados de realizar los cambios
  - deben informar acerca del estado de los cambios que están realizando
- El jefe de proyecto:
  - se debe asegurar de que los procedimientos de Control de Cambios se siguen correctamente.
  - Puede participar en la evaluación de los cambios, sobre todo en la determinación de impactos sobre otros elementos software, del coste del cambio y de su efecto sobre la programación del proyecto.
  - Informar acerca del estado de los cambios.
- El bibliotecario:
  - debe controlar la realización de los cambios sobre las últimas versiones (o las adecuadas, en todo caso)
  - Transferirá los elementos a modificar desde la Biblioteca de Soporte del Proyecto a la Biblioteca de Trabajo. La Biblioteca Maestra no se modificará hasta que el cambio se haya incorporado y probado.

En cualquier caso es necesario establecer de forma precisa, al comienzo de cada proyecto, cuál será el proceso de gestión de cambios que se va a utilizar. Para ello, será necesario definir:

- Políticas a nivel organizativo que promuevan las actividades de control de cambios.

- Los estándares que se van a adoptar y a los que será necesario ajustarse.

- Los procedimientos que se van a utilizar para poner en práctica las políticas de Gestión de Configuración.

Las políticas de Gestión de Configuración deben reflejar la filosofía y las metas de la organización en cuanto a las actividades de Gestión de Configuración.

Mientras que el objetivo de las políticas es definir el 'por qué' de la Gestión de Configuración, deben establecer un marco para definir el 'qué', el 'cuándo', el 'cómo' y el 'quién':

- Las responsabilidades.

- El contenido de las líneas base.

- Los tipos de cambios que se van a controlar.

- Las funciones del Comité de Control de Cambios.

- El flujo de documentación entre el solicitante de un cambio y el Comité de Control de Cambios.

- Los criterios para la valoración de las solicitudes de cambio, tanto de tipo técnico como de gestión.

- etc.

3.1. El proceso de control de cambios

No hay ningún estándar para el control informal o interno de los cambios, aunque sí hay algunas recomendaciones (IEEE STD 1042 Guide to Software Configuration Management).

En cuanto al control de cambios formal, se puede estructurar de muchas formas. Vamos a ver cuáles son las etapas típicas de un proceso formal, es decir, el proceso que habría que seguir para hacer un cambio sobre una línea base:

1. Iniciación del Cambio: se presenta una solicitud de cambio, que puede venir provocada por un problema que se ha detectado o por un cambio en los requisitos.

2. Clasificación y registro de la solicitud de cambio.

3. Aprobación o rechazo inicial de la solicitud de cambio. De ello suele ser responsable el *Comité de Control de Cambios* .

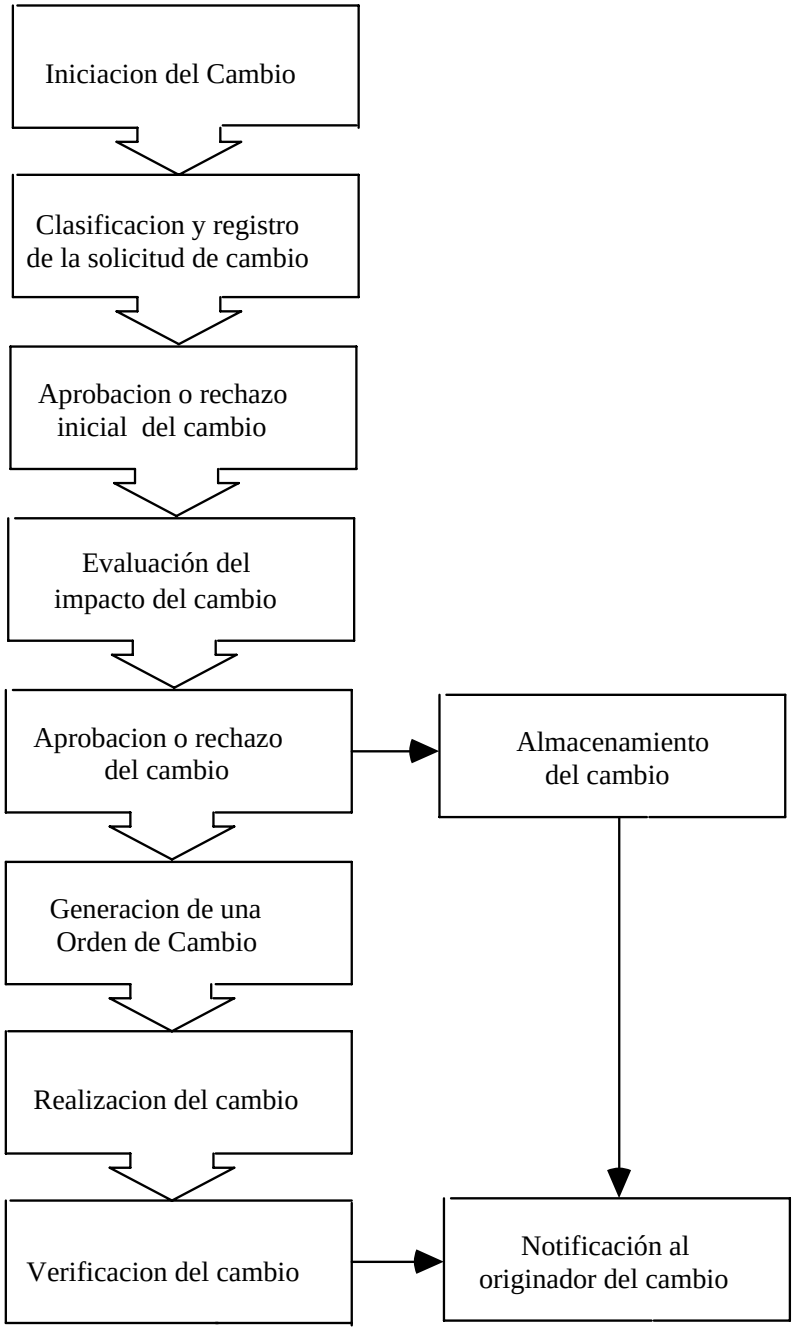
4. Evaluación de la solicitud de cambio, si ha sido aprobada, para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio. Como resultado se obtiene un *Informe de Cambio*.

5. Se presenta el Informe de Cambio al Comité de Control de Cambios. Si se considera que el cambio es beneficioso se genera una *Orden de Cambio* (también llamada Orden de Cambio de Ingeniería), que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría. Esta Orden de Cambio es asignada a alguno de los ingenieros de software para que se encargue de llevarlo a cabo. En este momento, el objeto a cambiar se da de baja en la Biblioteca de Soporte al Proyecto.

6. Se realiza el cambio, entrando en un proceso de seguimiento y control.

7. Una vez finalizado el cambio, se certifica, mediante una revisión, que se ha efectuado correctamente el cambio y con ello se ha corregido el problema detectado o bien se han satisfecho los requisitos modificados. El objeto se devuelve a la Biblioteca de Soporte al Proyecto.

8. Se notifica el resultado al originador del cambio.



En un proceso semi-formal, puede suprimirse la necesidad de generar la solicitud de cambio, el informe de cambio y la orden de cambio, pero sí debe realizarse la evaluación del cambio y su seguimiento.

En cualquier caso es necesario establecer de forma precisa, al comienzo de cada proyecto, cuál será el proceso de control de cambios que se va a utilizar.

Al definir este proceso, será también necesario:

- Definir los mecanismos para solicitar cambios sobre los Elementos de Configuración.

- Definir los mecanismos para analizar y evaluar el impacto de las solicitudes de cambio.

- Definir los mecanismos para aprobar o rechazar las solicitudes de cambio.

- Definir los mecanismos para controlar la realización de los cambios aprobados.

Los procesos de alta y baja de la Biblioteca del Proyecto implementan dos elementos importantes del Control de Cambios: el control de acceso y el control de sincronización:

- El Control de Acceso se refiere a los derechos que tienen los diferentes miembros del equipo de desarrollo para acceder y modificar ECS concretos. Así, por ejemplo, hay que controlar el acceso del ingeniero de software que da de baja el ECS de la Biblioteca de Proyecto para realizar un cambio aprobado por una Orden de Cambio.

- El Control de Sincronización ayuda a asegurar que los cambios en paralelo, realizados por equipos o personas diferentes, no se sobreescriben. Así, cuando un ECS se da de baja de la Biblioteca de Soporte, el Control de Sincronización bloquea el objeto para que no se puedan hacer más actualizaciones sobre él hasta que se haya reemplazado con una nueva versión.

El almacén de una herramienta de control de versiones se puede considerar como la Biblioteca de Soporte o de Proyecto. Estas herramientas ofrecen también de forma automática el control de acceso y control de sincronización.

3.2. Mecanismo para solicitar cambios

El primer mecanismo que es necesario definir es el mecanismo para solicitar cambios sobre los Elementos de Configuración.

El primer paso es definir el formulario que se debe utilizar para solicitar cambios. Este formulario de Solicitud de Cambio debe contener información suficiente para determinar:

- por qué se necesita el cambio (el elemento no funciona tal y como debe hacerlo, le falta alguna característica, etc.)

- qué hay que cambiar

- quién lo solicita

- descripción del problema lo suficientemente detallada como para que se pueda recomendar una solución.

- otros ECS que se pueden ver afectados por el cambio.

- otros cambios con los que está relacionado.

- aprobación del cambio.

- cómo se solucionó el problema, etc.

En cualquier caso, debe ser simple, y aceptado por las personas que lo tendrán que manejar.

El formulario más común es el de Solicitud de Cambios. Sin embargo, casi siempre se utilizan formularios diferentes para solicitar una mejora y para informar de un problema o deficiencia detectado durante una auditoría, una prueba o el uso del sistema (y que posiblemente requerirá un cambio).

A los formularios que se usan para informar de problemas se les suele llamar Informes de Incidencias o Informes de Problemas. Recogen información adicional sobre el incidente que ha desvelado la existencia de un problema:

- Fecha y hora en que ocurrió,

- descripción del incidente,

- efectos que ha producido,

- de qué forma se puede duplicar el incidente, si es que se ha podido duplicar,

- volcado de datos,

- referencia al tipo de prueba que se estaba efectuando.

Este Informe de Incidencia es analizado por los desarrolladores, y estos pueden recomendar alguna de las siguientes acciones:

- No requiere acción: Cuando lo que se describe en el informe de incidencia no es realmente una deficiencia. Esta situación suele ser debida a malentendidos acerca de la forma de funcionamiento del sistema. También se puede dar esta situación cuando ya se ha informado previamente de una indidencia similar, y se están tomando las acciones correctivas necesarias.

- Solicitud de Cambio: La implementación se corresponde con el diseño del sistema, pero una mejora en el diseño del sistema solucionaría el problema. Se genera entonces una Solicitud de Cambio, que se tratará por los cauces normales.

- Notificación de Cambio: Cuando la deficiencia que se describe en el informe de incidencia se debe a una mala implementación que debe ser corregida. Se informa entonces al CCC y se pasa a corregir la deficiencia.

Aparece, como vemos, un nuevo formulario, la Notificación de Cambio. Una Notificación de Cambio puede dar respuesta a varios Informes de Incidencias, cuando todos ellos corresponden al mismo problema.

Para solicitar cambios en la documentación se pueden utilizar los formularios anteriores o un nuevo formulario al que se suele llamar Formulario de Seguimiento de la Documentación. La resolución de un Informe de Incidencia o de una solicitud de

cambio también puede dar lugar a un formulario de este tipo, cuando la deficiencia tiene algún tipo de impacto sobre la documentación.

3.3. Mecanismo para aprobar o rechazar las solicitudes de cambio

Algunos criterios que se pueden tener en cuenta para tomar la decisión de aprobar o rechazar las solicitudes de cambio son los siguientes:

- Valor del cambio para el proyecto/organización
- Retorno de la inversión
- Tamaño
- complejidad
- impacto sobre el rendimiento del producto (uso de memoria y CPU)
- recursos disponibles para efectuar el cambio (humanos y materiales)
- relación con otros cambios ya aprobados y en progreso
- tiempo estimado para completar el cambio
- relación con las políticas de la empresa (satisfacción del cliente, competitividad, etc.)
- existencia de alternativas, etc.

3.4. Seguimiento de los cambios. Gestión de Problemas

Una vez aprobado un cambio debido a un problema se debe realizar un seguimiento del mismo. A este proceso se le llama Gestión de Problemas.

La Gestión de Problemas se considera una actividad complementaria a la de Control de Cambios, que consiste en gestionar la evolución de los problemas detectados sobre el software desarrollado, tanto aquellos que se detectan en la fase de pruebas como los informes de problemas que llegan del usuario.

Conlleva tareas como:

- Admisión y registro de notificaciones de problemas (via llamadas telefónicas, correo electrónico, herramienta específica)
- Asignación del problema a una persona responsable.
- Asociación de información al problema y mantenimiento de la misma en una Base de Datos de Problemas.

- Monitorización del estado del problema.
- Registro de actividades efectuadas para resolver un problema.
- Generación de informes acerca de los problemas.
- Resolución de interrogaciones sobre la Base de Datos de Problemas.
- Análisis estadísticos acerca de los problemas, como por ejemplo el estudio de correlaciones entre problemas y componentes.

Algunos de los datos que puede resultar interesante almacenar acerca de los problemas son:

- descripción,
- severidad,
- urgencia o prioridad
- causa del problema (omisiones en el análisis, error en la documentación de entrada, falta de experiencia,...)
- solución al problema
- módulos afectados,
- persona que lo notificó,
- persona responsable,
- fechas de notificación, resolución, etc.
- fase/etapa en la que se originó el problema
- fase/etapa en la que se detectó el problema

4. GENERACIÓN DE INFORMES DE ESTADO

Es la tercera de las cuatro tareas básicas de Gestión de Configuración. A veces también denominada Contabilidad de Estado. El objetivo es mantener a los usuarios, a los gestores y a los desarrolladores al tanto del estado de la configuración y su evolución. En definitiva, pretende dar respuesta a la pregunta “¿Qué ocurrió?”, y también a la pregunta “¿Cuándo ocurrió?”.

Ayuda también a mejorar los problemas de comunicación entre los participantes en un proyecto.

Esto se va a conseguir registrando toda la información necesaria acerca de lo que va ocurriendo y generando los informes necesarios. Esta tarea implica, por tanto, la realización de tres actividades básicas:

- Captura de la información

- Almacenamiento de la información

- Generación de informes

Así pues, los productos de esta actividad son fundamentalmente de dos categorías:

- Registros.

- Informes.

¿Por qué es importante esta tarea?

- Para mantener la continuidad del proyecto. Se trata de permitir que el proyecto siga adelante cuando, por ejemplo, el jefe de proyecto deja la empresa.

- Para evitar la duplicidad del trabajo. Si no se guarda información acerca de lo que ya se ha hecho, se puede estar repitiendo el trabajo ya hecho.

- Para evitar caer en los mismos errores una y otra vez.

- Para ser capaces de repetir aquello que salió bien.

- Puede ayudar a encontrar las causas de un fallo.

Centrándonos en el capítulo de Captura de la información, cabe preguntarse ¿de dónde proviene la información necesaria para poder realizar esta tarea? Básicamente, la información proviene de las otras tres actividades de GCS. Y entonces ¿qué información interesa capturar? La cantidad y tipo de información a capturar depende de las características del proyecto, de su tamaño y complejidad. Sin embargo, al comenzar un proyecto nunca se sabe qué información puede llegar a ser útil y cuál no. Por eso, es preferible guardar la mayor información posible. En cualquier caso, el Plan de Gestión de Configuración deberá indicar la información mínima a capturar.

Una vez se ha decidido qué información capturar hay que decidir dónde almacenarla. Lo mejor suele ser almacenarla en una base de datos y utilizar herramientas automatizadas para gestionarla.

La información contenida en esta base de datos se podrá utilizar, además de para generar los informes de Contabilidad de Estado pertinentes, para facilitar otras tareas como:

- El análisis post-mortem del proyecto

- La realización de estimaciones para proyectos futuros

4.1. Registros

Algunos ejemplos de los tipos de registros que pueden mantenerse son los siguientes:

a) Registro de elementos de configuración: Conteniendo toda la información relativa a los diferentes elementos de configuración.

b) Registro de Líneas base: Conteniendo toda la información relativa a cada línea base:

Nombre

Fecha de establecimiento

Elementos de configuración que la componen

c) Registro de Solicitudes de cambios: El tipo de información que se suele mantener acerca de cada solicitud de cambio es la recogida a través del formulario de Solicitud de Cambio, incluyendo:

- Código de solicitud

- Información acerca del solicitante

- descripción del cambio

- la documentación que apoya la petición de cambio, por ejemplo, una referencia a un Informe de Incidencia.

- la resolución o disposición acerca del cambio (aprobado, aplazado, denegado,...)

d) Registro de Cambios: El tipo de información que se suele mantener acerca de cada cambio es la recogida a través de el Informe de Cambio, la Orden de Cambio, el proceso de Gestión de Problemas, etc. Puede contener información acerca de:

1. Solicitud del cambio a la que corresponde

2. Evaluación del cambio:

    - Coste

    - Esfuerzo

    - Tiempo

    - Soluciones alternativas

5. Plan de implementación del cambio.

6. Restricciones y criterios de revisión

7. Impacto sobre la configuración:

- Líneas base afectadas

- Elementos de configuración afectados

- Versiones afectadas

8. Historia del cambio: Puesto que un cambio es algo que evoluciona, es necesario mantener una historia de cada cambio. En cuanto a los datos que se deben mantener en la historia de un cambio, se pueden considerar:

- Fecha de la solicitud de cambio.

- Fecha de aprobación del cambio.

- Fecha de rechazo del cambio.

- Fecha de cancelación del cambio.

- Fecha de implementación del cambio.

- Fecha de cierre del cambio.

etc.

e) Registro de Incidencias: El tipo de información que se suele mantener acerca de cada incidencia es la recogida a través del Informe de Incidencia, del tipo:

1. Información del incidente

2. Resultado de la Incidencia

- Disposición del CCC

- Número de la solicitud de cambio a la que dió lugar (si es aplicable)

- Número de Formulario de Seguimiento de Documentación (si es aplicable)

- Número de Notificación de Cambio asignada (si es aplicable)

3. Historia:

- Fecha de la incidencia.

- Fecha de resolución acerca de la incidencia

- Fecha de cierre de la incidencia

f) Registro de Modificaciones del Código: Puede contener información del tipo:

- Número de identificación de la modificación

- Descripción de la modificación
- Número de notificación de cambio a la que corresponde (si es aplicable)
- Número de solicitud de cambio a la que corresponde (si es aplicable)
- Nombre de los módulos afectados
- Versiones modificadas
- Persona responsable de la modificación
- Fecha de inicio
- Fecha de terminación
- etc.

g) Registro de Modificaciones sobre bases de datos:

- Número de identificación de la modificación
- Descripción de la modificación
- Número de notificación de cambio a la que corresponde (si es aplicable)
- Número de solicitud de cambio a la que corresponde (si es aplicable)
- Base de Datos modificada
- Número de versión modificada
- Registros modificados
- Persona responsable de la modificación
- Fecha de inicio
- Fecha de terminación
- etc.

h) Registro de Modificaciones sobre documentación:

- Número de identificación de la modificación.
- Descripción de la modificación
- Número de formulario de seguimiento de documentación al que corresponde
- Documento modificado
- Número de versión modificada

- Persona responsable de la modificación

- Fecha de inicio

- Fecha de terminación

- etc.

i) Registro de Releases y Variantes: Su objetivo es describir la composición y estado de una versión liberada del producto:

- Código de release o variante

- Fecha de liberación

- Elementos de configuración que la componen

- Versión de los elementos de configuración que la componen

- Medio en el que se encuentran

- Diferencias con la release anterior

    - Cambios incorporados

    - Cambios pendientes

j) Registro de Instalaciones: Su objetivo es mantener información acerca de todos los lugares en los que se ha instalado un producto software. Puede contener información del tipo:

- Identificación del producto

- Lugar en el que se ha instalado

- Fecha de la instalación

- Release instalada

k) Actas de las reuniones del Comité de Control de Cambios:

1. Fecha

2. Lista de miembros asistentes

3. Propósito de la reunión

4. Acciones del CCC:

    - ECS etiquetados

    - Líneas base revisadas/cambiadas

- Disposición de Solicitudes de Cambio, Informes de Incidencias, Notificaciones de Cambio, Formularios de Seguimiento de Documentación, Informes de Cambios, etc.

- Líneas base aprobadas

5. Resultados de las auditorías:

- Deficiencias detectadas

- Plan de resolución de las deficiencias encontradas

- Recomendaciones

4.2. Informes

En cuanto a los informes, podemos distinguir dos tipos:

- Planificados
- Bajo demanda

Algunos ejemplos de los tipos de informes que se pueden generar son:

a) Informe de estado de los cambios: Es un resumen del estado en que se encuentran todas las solicitudes de cambio registradas durante un determinado período de tiempo.

b) Inventario de elementos de configuración, para ofrecer visibilidad sobre el contenido de las bibliotecas de proyecto.

c) Informe de incidencias: Es un resumen del estado en que se encuentran todas las incidencias originadas durante un determinado período de tiempo y las acciones a las que han dado lugar.

d) Informe de modificaciones: Es un resumen de las modificaciones que se han efectuado en el producto software durante un determinado período de tiempo.

e) Informe de diferencias entre versiones: Resumen de las diferencias entre las sucesivas versiones de un elemento de configuración.

En cualquier caso, al comienzo de cada proyecto será necesario decidir qué tipo de registros se van a mantener y qué tipo de informes se van a generar y para quién.

5. AUDITORÍA DE LA CONFIGURACIÓN

Una auditoria es una verificación independiente de un trabajo o del resultado de un trabajo o grupo de trabajos para evaluar su conformidad respecto de especificaciones,

estándares, acuerdos contractuales u otros criterios. La auditoria de la Configuración es la forma de comprobar que efectivamente el producto que se está construyendo es lo que pretende ser.

Esta función a veces se considera fuera de la Gestión de Configuración y dentro de la Garantía de Calidad. También tiene relación con las actividades de Validación y Verificación. En realidad es un punto de intersección entre todas ellas.

Es la actividad de GCS más costosa. Requiere de personal experimentado, y con un gran conocimiento del proceso de desarrollo. Sin embargo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad de la auditoria.

Se pueden diferenciar tres tipos de actividades:

- Revisiones de fase: Se realizan al finalizar cada fase del desarrollo y su objetivo es examinar los productos de dicha fase. Las revisiones propias de la Gestión de configuración son aquellas en las que se establecerán las líneas base. El objetivo de esta revisión es descubrir problemas, no comprobar que todo está bien. Hay que ser capaz de desenmascarar los problemas ocultos y sutiles, no sólo los que son obvios.

- Revisiones de cambios: Se realizan para comprobar que los cambios aprobados sobre una línea base se han realizado correctamente.

- Auditorías: Se realizan al final del proceso de desarrollo de software y su objetivo es examinar el producto en su conjunto.

Las revisiones se deben realizar de forma continua, durante todo el proceso de desarrollo, y no sólo al finalizar éste, cuando los problemas ya no tienen solución.

La tarea de revisión implica tres tipos de funciones:

- Verificar que la configuración actual del software se corresponde con lo que era en fases anteriores. Debe haber correspondencia y trazabilidad entre los elementos de configuración que aparecen en una línea base y los que aparecen en las línea base que la preceden y que la siguen. La verificación se realiza con respecto a la línea base precedente.

- Validar que la configuración actual del software satisface la función que se esperaba del producto en cada hito del proceso de desarrollo. La validación se realiza con respecto a los requisitos del sistema.

- Valorar si una determinada línea base, teniendo en cuenta los resultados de la verificación y validación, y otro tipo de comprobaciones, se debe considerar aceptable o no.

El papel que juegan las revisiones en el proceso de Gestión de Configuración es el siguiente:

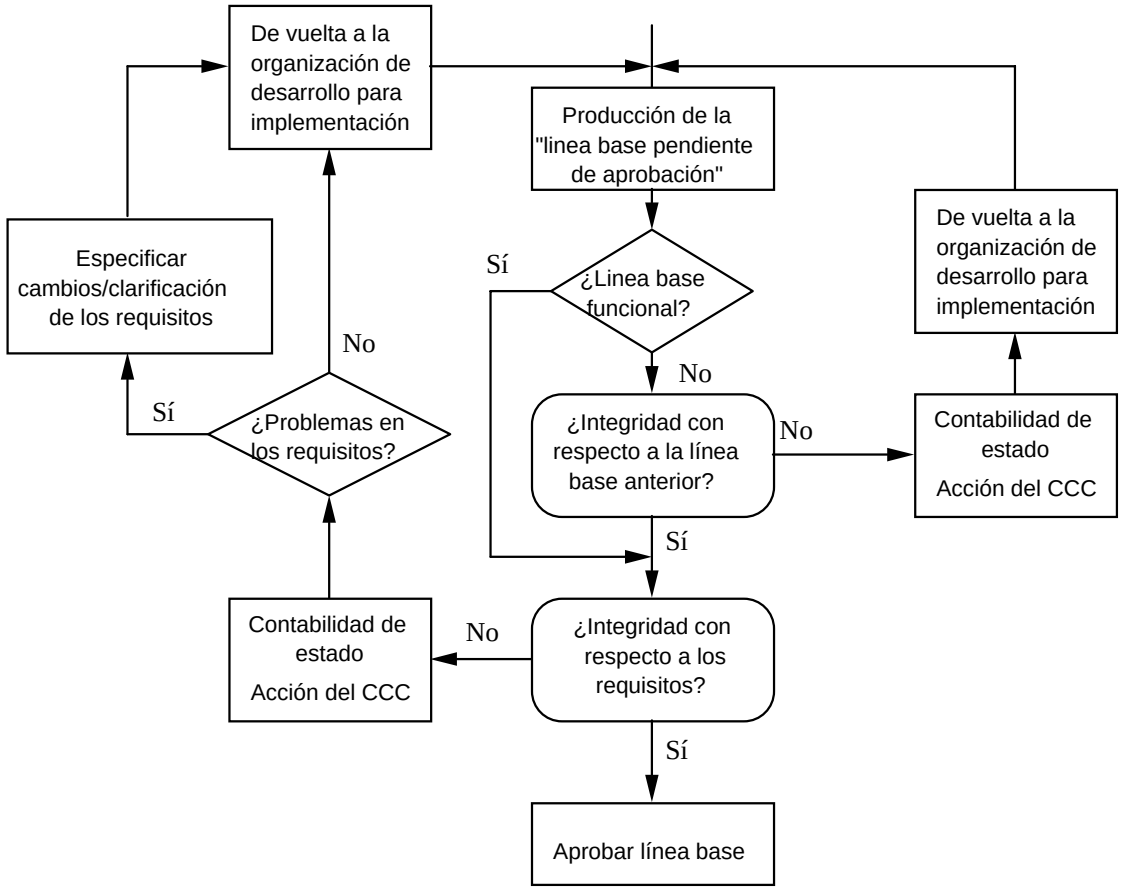
1. Los productos generados durante el proceso de desarrollo se agrupan, al llegar determinados hitos, en una “línea base pendiente de aprobación”.

2. Tiene lugar la revisión de fase

3. Si en la revisión se encuentran deficiencias en los ECS que componen la línea base, se generan los correspondientes Informes de Problemas, o Informes de Discrepancias, y se entregan al CCC, el cual recomienda ciertos Cambios sobre la Configuración.

4. Una vez implementados los cambios propuestos, se pasa a una nueva revisión de cambios en la que se comprueba si los cambios se han implementado correctamente.

5. Si en la revisión no se encuentra ninguna deficiencia, se declara “aceptable” la “línea base pendiente de aprobación”. Si el CCC está de acuerdo con la resolución de los revisores, la línea base se aprueba.



En cuanto a las auditorías, se suelen distinguir dos tipos de auditorías de configuración:

- Auditoría Funcional: Cuyo objetivo es comprobar que se han completado todos los tests necesarios para el Elemento de Configuración auditado, y que, teniendo en cuenta los resultados obtenidos en los test, se puede afirmar que el Elemento de Configuración satisface los requisitos que se impusieron sobre él.

- Auditoría Física: Cuyo objetivo es verificar la adecuación, completitud y precisión de la documentación que constituye las líneas base de diseño y de producto. Se trata de asegurar que representa el software que se ha codificado

y probado. Tras la auditoría física se establece la línea base de Producto. Tiene lugar inmediatamente después de haberse superado la auditoría Funcional.

Y aún se puede considerar un tercer tipo de auditoría:

- Revisión Formal de Certificación: Cuyo objetivo es certificar que el Elemento de Configuración del Software se comporta correctamente una vez que éste se encuentra en su entorno operativo.